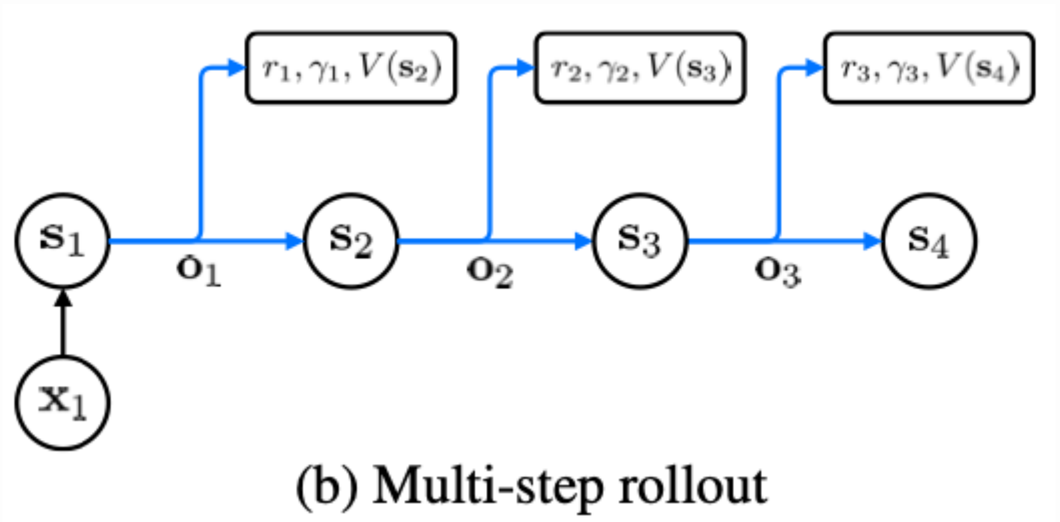
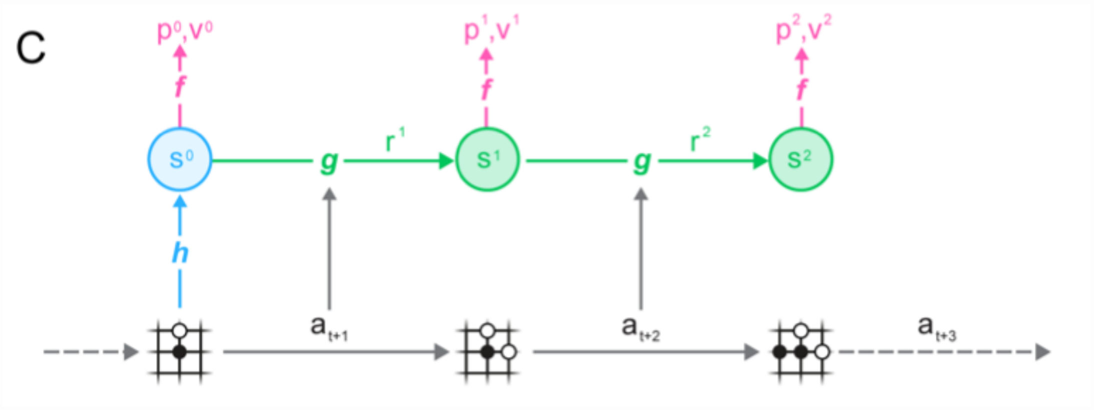


From Value Prediction Network (2017)



From MuZero (2019)

You have three functions. MuZero calls them h (representation), g (dynamics), and f (prediction).

A **representation** function takes in the sensor data and transforms the manifold to something more learnable, referred to as the “hidden state”. It may discard non task relevant information. This can be a VAE, GAN, pretrained ImageNet, custom trained conv net, or in the case of GPT-3, handcoded as BPE.

A **dynamics** function deals with long term temporal correspondences. It functions as both a summarizer of the past and a predictor of the future, in the same way a Kalman filter does. This can be a couple dense layers, a transformer, or an RNN/GRU/LSTM. I guess it even could be a Kalman filter itself.

A **prediction** function tells you how to act in a state. It can do this by directly outputting an action or by outputting a value of this state allowing you to search.

(update in 2023: you also need a **critic** function)

I'm the biggest shill for learning everything. There's nothing about traffic cones in your DNA, so if you are trying to build a driving agent, I don't think hiring a guy to write cone handling software is very smart. At comma, we try to learn the same stuff the human does, while handcoding the same stuff that's coded in human DNA.

There's two papers I love that illustrate what understanding a system really means. Can a biologist fix a radio? (2002) and Could a Neuroscientist Understand a Microprocessor? (2017). To me, they illustrate that the current tools of biology and subfields are unequipped to handle understanding life. We will know we understand life once we can build life. We will know we understand a brain once we can build a brain.

I think it's going to end up looking like birds and planes. Planes fly in a much more "rigid" way, and they are much simpler than birds. But for our purposes, planes are more useful, and nobody would deny that they are "flying machines"

Nobody will deny that our artificial life are "thinking machines." Nobody will care if the machine can "love" in the same way nobody cares if a plane eats fish. We will care if we've built something that can do everything useful that a human can do.

"Everything should be made as simple as possible, but no simpler."

These three fundamental functions are here to stay. I believe once we understand the DNA coding for the brain, we will find things that look like these three in there.

I have a far better intuition for how the brain works from studying machine learning than I ever got from neuroscience or cognitive psychology. Just as building model planes taught me more about flight than birds.

And to those who say, but *the metaphors of the age are always used to describe the brain*. Aka in the 1800s it was a steampunk brain, in the 1900s it was a calculator, and in the 2000s it's a computer. To them I say, **watch us build a brain, watch us build life**. No other era did, we will. Because we are starting to understand.

And you'll forget about that Chinese room so fast, when this "machine life" is cooking and cleaning and driving you around and making you fall in love with it. Of course it's in love with you too. I mean, you feel like it is, right? There's no such thing as consciousness, we are all p-zombies, and philosophy is for high schoolers who just discovered smoking weed.

Specifying the network is the easier part. The harder part is the training algorithm. This is what we do at comma.

Our **representation** function, which we call the "vision model" is an EfficientNet-B2 (2019). It takes in 2 YUV frames and outputs a 1024 dimensional vector that covers a gaussian space due to being trained with a KL loss term. While it could be only one frame, we argue that 2 is the fundamental sensor input to allow the sensor to perceive motion. It could even be k frames, just not n :)

Our **dynamics** function, which we call the "temporal model" is a GRU. And our **prediction** function is just a couple of dense layers on the GRU output.

We first jointly train our **representation** and **prediction** (thrown away) in a supervised manner on the path the human drove. While we could also jointly train the **dynamics** here, and it would likely be slightly better at understanding temporal relations, for compute reasons we don't.

We freeze the weights of the **representation**, then train **dynamics** and **prediction** in a simulator. Supervised learning from the human policy doesn't work due to behavioral cloning, these slides explain the problem well. We have more info about the training in this blog post.

It's possible to learn all three functions jointly using full RL, both VPN and MuZero do. But this assumes you have access to both a simulator and a reward function. In the real world, you don't have either.

While we use a simulator, we aren't actually doing RL with a reward function. Even in the simulator the ground truth is the human path, it just doesn't always start at the center of the car.

While MuZero is making less calls to the simulator than AlphaZero (2017), it still has to act and get on-policy feedback sometimes. At comma, we are going to deal with this by doing *reinforcement learning on the world*, but with the temperature parameter at 0. Nobody would be happy if their car "explored", it should only "exploit". A 0 temperature parameter is okay for learning, but only if your network is close to converged, hence why we'll be keeping around the current training approach for pretraining.

Now, you say, but you do have a simulator! You talked about it, why can't you just use MuZero and call it a day. There's two issues:

One, the simulator is incomplete. The hardest part of any simulator is the behavior of the other agents, and in ours they are fixed to the behavior of what the cars actually did on the route we are simulating. While at least they were driven with a human policy (god help those who think they can learn to drive in some Unreal Engine environment with a hacked together game like car AI), their policy rollout can't change in response to your actions. Aka it can't handle counterfactuals.

Two, even if we had a perfect simulator, we don't have a reward function. In the real world, the user taking over control can be viewed as negative reward. But in the simulator we don't have that. While it may be possible to learn a reward function on data from when users are likely to disengage, that function would have to be updated frequently as the machine policy changes.

We are very fortunate in driving that:

- the output space is low dimensional, so simple loss functions work pretty well
- it's easy to gather huge datasets of expert policy from humans
- there's a clear reward signal in L2 driving of human disengagements
- the car is already quite digitized, once you know the actions you can do them

I still believe driving is the best applied AI problem today. It's simpler than cleaning a house, but if you solve it in the right way a lot of the parts will be